Document Title	Specification of COM Based Transformer
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	662

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R24-11

Document Change History

Date	Release	Changed by	Description
2024-11-27	R24-11	AUTOSAR Release Management	• Changed Reentrancy of Init and DeInit.
2023-11-23	R23-11	AUTOSAR Release Management	• No content changes
2022-11-24	R22-11	AUTOSAR Release Management	• No content changes
2021-11-25	R21-11	AUTOSAR Release Management	• Updated buffer reservation in transformer chain
2020-11-30	R20-11	AUTOSAR Release Management	• No content changes
2019-11-28	R19-11	AUTOSAR Release Management	• editorial • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Updated buffer handling • Removed include file structure
2017-12-08	4.3.1	AUTOSAR Release Management	• Updated handling of gaps in the array representation of a signal group. • Clarification on parameter passing.

▽

△

2016-11-30	4.3.0	AUTOSAR Release Management	• Updated include file structure figure. • Clarification on postBuild configuration in chapter 10. • Added support for unqueued communication when no data is available in [SWS_ComXf_00035].
2015-07-31	4.2.2	AUTOSAR Release Management	• Exclude support for external trigger communication [SWS_ComXf_00032]
2014-10-31	4.2.1	AUTOSAR Release Management	• Initial Release

Contents

1 Introduction and functional overview

The transformer feature provides functionality to chain arbitrary transformers when sending and receiving data in the RTE. The COM Based Transformer provides this functionality when the target bus system uses a fixed communication matrix with packed data representations.

2 Acronyms and Abbreviations

No specific terms have been introduced additionally to those already defined in [1].

3 Related documentation

3.1 Input documents

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] General Specification of Transformers
AUTOSAR_CP_ASWS_TransformerGeneral

[3] Specification of RTE Software
AUTOSAR_CP_SWS_RTE

[4] Specification of Communication
AUTOSAR_CP_SWS_COM

[5] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

[6] Requirements on Transformer
AUTOSAR_CP_RS_Transformer

[7] System Template
AUTOSAR_CP_TPS_SystemTemplate

[8] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

3.2 Related standards and norms

Not applicable.

3.3 Related specification

AUTOSAR provides a General Specification on Transformers [2], which is also valid for COM Based Transformer.

Thus, the specification ASWS Transformer General shall be considered as additional and required specification for COM Based Transformer.

4 Constraints and assumptions

4.1 Limitations

For the COM Based Transformer all general transformer limitations (see [2]) apply.

Additionally the following restrictions apply for the COM Based Transformer:

[SWS_ComXf_00017]

Upstream requirements: SRS_Xfrm_00201

⌈The COM Based Transformer does not support Client-Server communication.⌋

[SWS_ComXf_00032]

Upstream requirements: SRS_Xfrm_00201

⌈The COM Based Transformer does not support external trigger communication.⌋

[SWS_ComXf_00018]

Upstream requirements: SRS_Xfrm_00201

⌈The COM Based Transformer only supports composite data types (i.e. Signal Groups from COM).⌋

If the use-case occurs that a single primitive data element shall be handled by the COM Based Transformer (because there shall be a E2E protection performed after the serialization) the data element shall be wrapped in a structure. The structure would then just contain one entry, the signal group on COM level would also have the E2E parts included.

[SWS_ComXf_00019]

Upstream requirements: SRS_Xfrm_00201

⌈The COM Based Transformer only supports fix sized data types.⌋

[SWS_ComXf_00022]

Upstream requirements: SRS_Xfrm_00201

⌈The COM Based Transformer only supports signal groups which are byte aligned.⌋

[SWS_ComXf_00023]

Upstream requirements: SRS_Xfrm_00201

⌈The COM Based Transformer shall support signal group where all group signals are mapped successively (possibly with gaps where positions in the signal group layout have no corresponding signal defined) to the IPdu.⌋

4.2 Applicability to car domains

The COM Based Transformer can be used for all domain applications when a fixed communication matrix is used.

5 Dependencies to other modules

The AUTOSAR RTE [3] has to exist to execute the COM Based Transformer.

The AUTOSAR COM configuration [4] of the data handled by the COM Based Transformer has to exists in order to allow the configuration of the COM Based Transformer.

5.1 File structure

5.1.1 Code file structure

The source code file structure is defined in the [2].

6 Requirements Tracing

The following table references the features specified in [5] and [6] and links to the fulfillments of these.

Requirement	Description	Satisfied by
[SRS_BSW_00159]	All modules of the AUTOSAR Basic Software shall support a tool based configuration	[SWS_ComXf_00025]
[SRS_BSW_00337]	Classification of development errors	[SWS_ComXf_00028]
[SRS_BSW_00402]	Each module shall provide version information	[SWS_ComXf_91000]
[SRS_BSW_00404]	BSW Modules shall support post-build configuration	[SWS_ComXf_00030]
[SRS_BSW_00407]	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	[SWS_ComXf_00024] [SWS_ComXf_00026] [SWS_ComXf_00027]
[SRS_BSW_00411]	All AUTOSAR Basic Software Modules shall apply a naming rule for enabling/disabling the existence of the API	[SWS_ComXf_00024] [SWS_ComXf_00026] [SWS_ComXf_00027]
[SRS_BSW_00441]	Naming convention for type, macro and function	[SWS_ComXf_00030]
[SRS_Xfrm_00009]	A fixed set of transformer classes shall exist	[SWS_ComXf_00003]
[SRS_Xfrm_00011]	A transformer shall belong to a specific transformer class	[SWS_ComXf_00003] [SWS_ComXf_00006]
[SRS_Xfrm_00201]	The COM Based Transformer shall define the serialization of atomic and structured data elements into linear arrays based on a fixed data mapping	[SWS_ComXf_00004] [SWS_ComXf_00005] [SWS_ComXf_00007] [SWS_ComXf_00008] [SWS_ComXf_00009] [SWS_ComXf_00010] [SWS_ComXf_00011] [SWS_ComXf_00012] [SWS_ComXf_00013] [SWS_ComXf_00015] [SWS_ComXf_00016] [SWS_ComXf_00017] [SWS_ComXf_00018] [SWS_ComXf_00019] [SWS_ComXf_00020] [SWS_ComXf_00021] [SWS_ComXf_00022] [SWS_ComXf_00023] [SWS_ComXf_00032] [SWS_ComXf_00035] [SWS_ComXf_00036] [SWS_ComXf_00037]
[SRS_Xfrm_00202]	The COM Based Transformer shall take its configuration from the COM module	[SWS_ComXf_00005] [SWS_ComXf_00020] [SWS_ComXf_00025] [SWS_ComXf_00031] [SWS_ComXf_00033] [SWS_ComXf_00034] [SWS_ComXf_00036] [SWS_ComXf_00037]

Table 6.1: Requirements Tracing

7 Functional specification

When a SWC initiates an inter-ECU communication which is configured to be transformed, the SWC hands the data over to the RTE. The RTE executes the configured transformer chain which contains - if the configuration demands this - the COM Based Transformer.

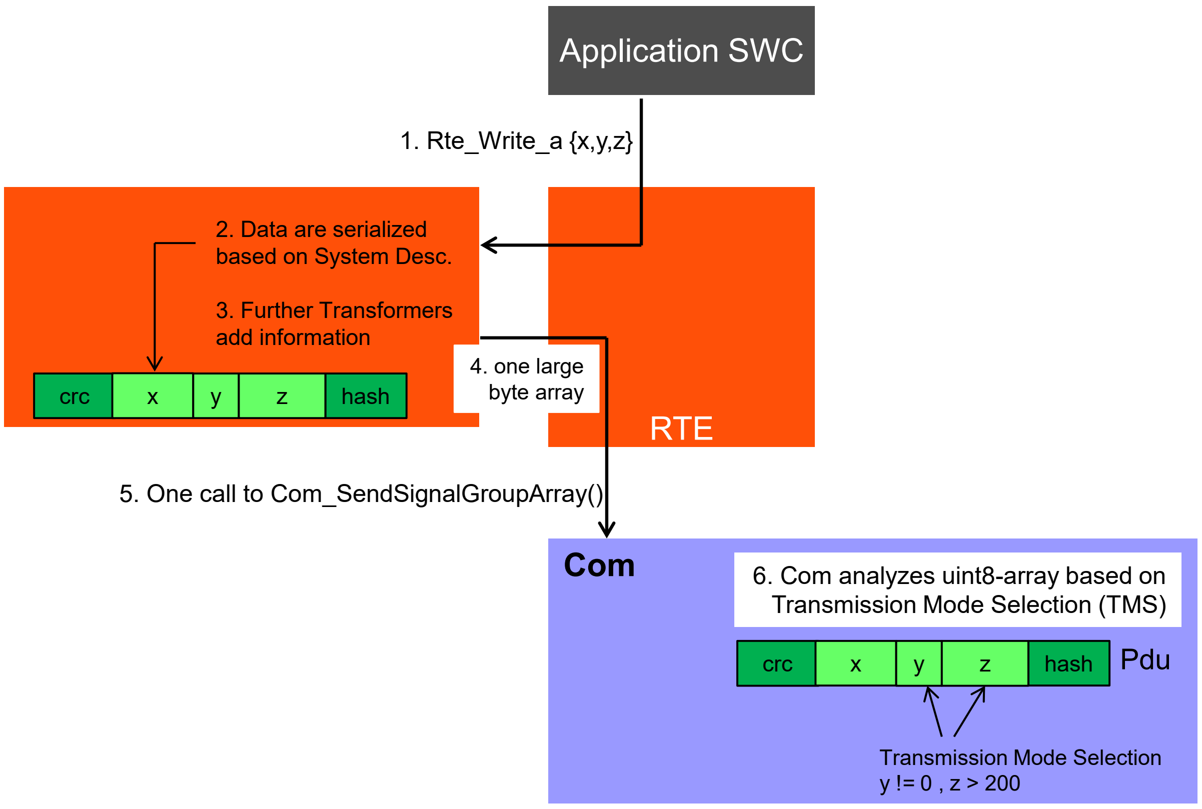
Application SWC
1. Rte_Write_a {x,y,z}
2. Data are serialized based on System Desc.
3. Further Transformers add information
crc
x
y
z
hash
4. one large byte array
RTE
5. One call to Com_SendSignalGroupArray()
Com
6. Com analyzes uint8-array based on Transmission Mode Selection (TMS)
crc
x
y
z
hash
Pdu
Transmission Mode Selection
y != 0 , z > 200

Figure 7.1: Overview of COM Based Transformer

The COM Based Transformer on the sender side serializes the data of the SWC and brings them into a uint8-array representation based on the communication matrix description. The uint8-array representation is forwarded to the COM module to be placed inside the respective IPdu. The COM module may analyze (depending on the configuration of the Transmission Mode Selection – TMS) the provided uint8-array and trigger the respective transmission mode. The IPdu is sent via the communication stack over the bus to the receiver(s).

The RTE of the receiver side executes the transformer chain in the reverse order. The COM Based Transformer of the receiver deserializes the linear data back into the original data structure. These are handed over to the receiving SWC.

From the SWC's point of view it is totally transparent whether data are transformed or not.

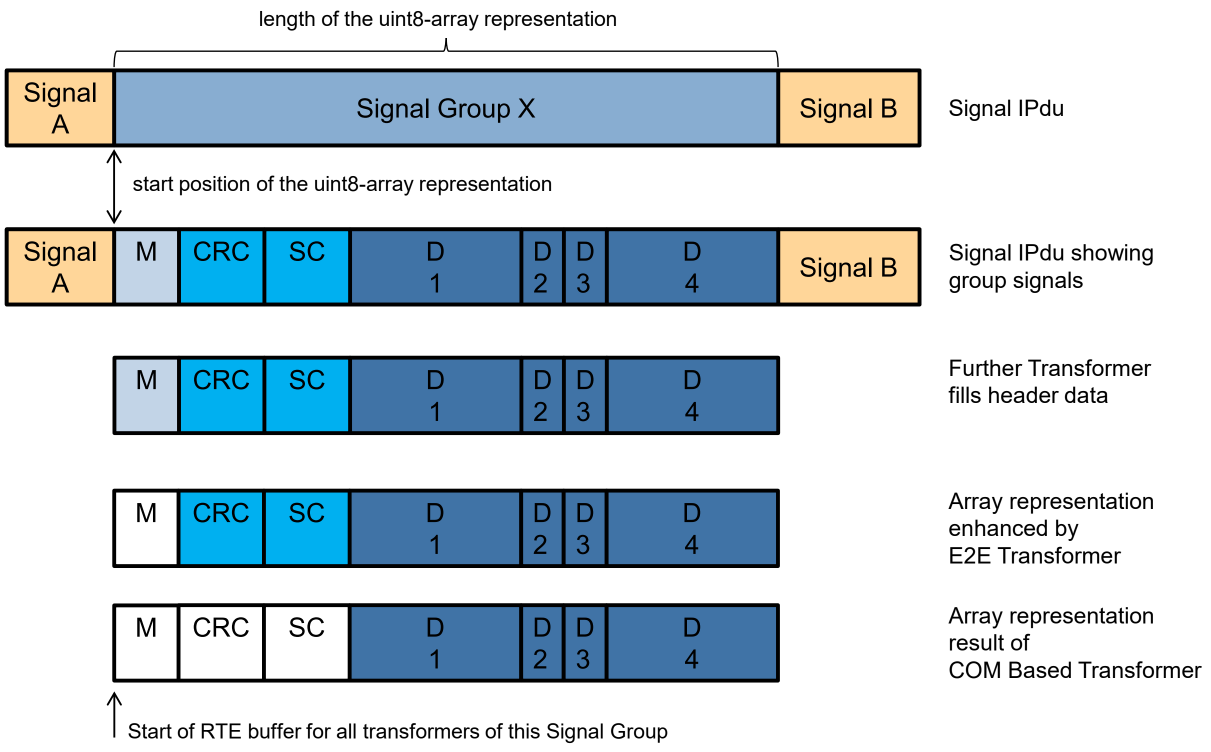
length of the uint8-array representation
Signal A
Signal Group X
Signal B
Signal IPdu
start position of the uint8-array representation
M
CRC
SC
D 1
D 2
D 3
D 4
Signal IPdu showing group signals
Further Transformer fills header data
Array representation enhanced by E2E Transformer
Array representation result of COM Based Transformer
Start of RTE buffer for all transformers of this Signal Group

Figure 7.2: IPdu and signal layout

The handling of the data inside COM's IPdu buffer and the transformer buffer is shown in figure 7.2.

The `ISignalIPdu` is handled by the COM module as `ComIPdu` and may contain several parts (signals and signal groups). A signal group in COM is represented by the `ComSignalGroup` container.

In the System Template [7] it is possible to define that a signal group shall be handled by the COM Based Transformer. The usage of the COM Based Transformer for a specific transformer chain is defined by the reference `comBasedSignalGroupTransformation` from the `ISignalGroup` to `DataTransformation` (see figure 7.3).

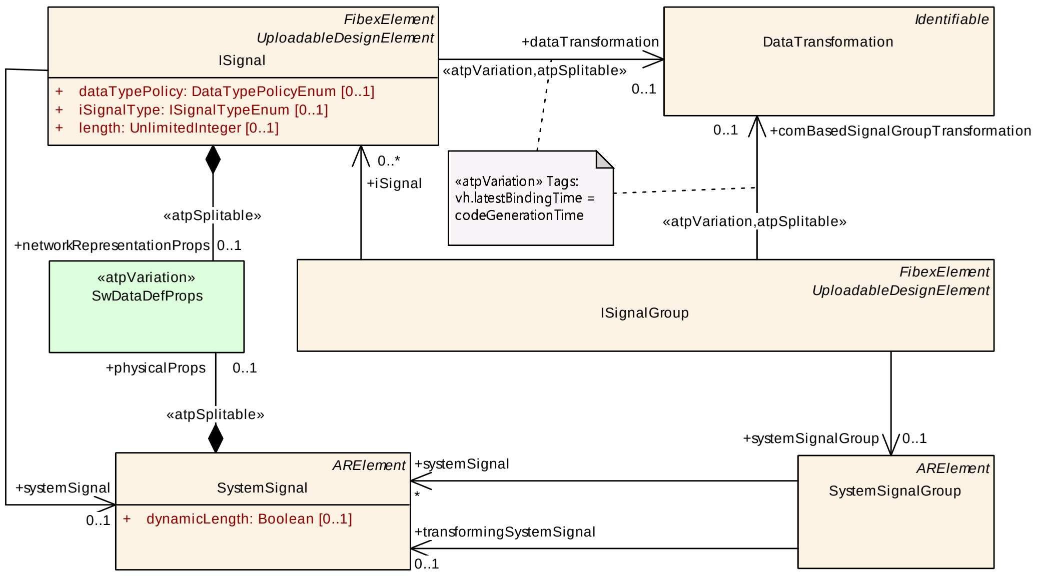

Figure 7.3: System Template Transformed communication

In the Ecu configuration of the COM module such ComSignalGroups have a ComSignalGroupArrayAccess=true parameter defined.

If the ComSignalGroupArrayAccess=true then it implicitly defines the length of the signal group ([SWS_Com_00845] [4]) and the start position inside the ComIPdu where the signal group starts ([SWS_Com_00844] [4]). Thus there can be several signals and signal groups defined inside an ISignalIPdu (e.g. 'Signal A' and 'Signal B' are part of the IPdu but are not considered by the COM Based Transformer for the 'Signal Group X').

The Com APIs Com_SendSignalGroupArray and Com_ReceiveSignalGroupArray handle the signal group as array representation based on length of the signal group and the start position inside the ComIPdu. This array representation contains all signals that belong to the signal group, regardless whether the application software has a data mapping defined or not.

As an example in figure 7.2 the 'Signal Group X' consists of the signals 'M', 'CRC', 'SC', 'D1', 'D2', 'D3', and 'D4'. Thus the RTE will interact with COM based on the whole array representation of this signal group with length of the signal group.

When the RTE interacts with the COM Based Transformer also all the other potential transformers need to be considered in order to determine which part of the array representation of the signal group actually is provided to each transformer since each transformer may add data during sending (or remove data during reception).

E.g. the part of the array representation which holds the 'CRC' and 'SC' signals are handled by the E2E transformer but will not be considered by the COM Based Transformer. The 'M' signal will neither be handled by the E2E transformer nor the COM Based Transformer. For 'M' a further transformer is responsible.

The RTE will provide the data buffer beginning at the 'Start of RTE buffer for all transformers of this Signal Group' location to the COM Based Transformer, the E2E Transformer, and any further transformer defined.

7.1 Specification of the COM Based Transformer

Serialization describes the way data is represented in protocol data units (PDUs) transported over a network. For the COM Based Transformer the serialization is defined by the communication matrix using the System Template [7]. The communication matrix information is taken over to the Ecu configuration of the COM module.

[SWS_ComXf_00005]

Upstream requirements: SRS_Xfrm_00201, SRS_Xfrm_00202

⌈The serialization is based on the Ecu configuration of the COM module and

- the Software Components `PortPrototype`,
- the `dataElement` list defined by the respective `SenderReceiverInterface`,
- the `SenderReceiverToSignalGroupMapping`,
- and the `ISignalToIPduMapping` for the `SystemSignalGroup`.

⌋

[SWS_ComXf_00003]

Upstream requirements: SRS_Xfrm_00009, SRS_Xfrm_00011

⌈The COM Based transformer shall only be used as the top-most transformer (first) in a transformer chain.⌋

The COM Based Transformer serializes structured data into a linear form. Therefore it can only be used as the first transformer on the sending side and the last transformer on the receiving side.

[SWS_ComXf_00004]

Upstream requirements: SRS_Xfrm_00201

⌈The COM Based transformer defined in this document shall be used as a transformer if

- the attribute `protocol` of the `TransformationTechnology` is set to `COM-Based`
- and the attribute `version` of the `TransformationTechnology` is set to `1.0.0`

- and the attribute transformerClass of the TransformationTechnology is set to serializer

⌋

[SWS_ComXf_00015]

Upstream requirements: SRS_Xfrm_00201

⌈The COM Based Transformer shall support all basic data types that are supported by the COM module in [SWS_Com_00675] except for UINT8_DYN.⌋

[SWS_ComXf_00016]

Upstream requirements: SRS_Xfrm_00201

⌈The COM Based Transformer shall handle each dataElement of the SenderReceiverInterface individually.⌋

[SWS_ComXf_00021]

Upstream requirements: SRS_Xfrm_00201

⌈The COM Based Transformer shall handle each dataElement of the SenderReceiverInterface like defined for the COM module [4] when the COM API Com_SendSignal (rep. Com_ReceiveSignal) is called for a shadow signal.⌋

This defines that the COM Based Transformer performs all actions equally to the COM module. This does include functionality like endianess conversion and sign extension.

The COM Configuration implicitly defines the length of the signal group ([SWS_Com_-00845] [4]) and the start position inside the ComIPdu where the signal group starts ([SWS_Com_00844] [4]). In order to place the transformed data element into the data buffer provided by the RTE the COM Based Transformer needs to respect the offset introduced by the position of the ComGroupSignal inside the ComIPdu (defined by the start position inside the ComIPdu) and the additional offset introduced by header data which is handled by other transformers called after the COM Based Transformer.

[SWS_ComXf_00036]

Upstream requirements: SRS_Xfrm_00201, SRS_Xfrm_00202

⌈If the signal layout of the signal group array representation contains gaps, those gaps shall be set during transmission to the value defined by the ComTxIPduUnusedAreasDefault of the respective ComTxIPdu that this signal group is mapped to.⌋

Gaps in the signal group array representation may occur because the layout is not fully packed and there are bits (or even bytes) that have no signal defined for (see 7.4).

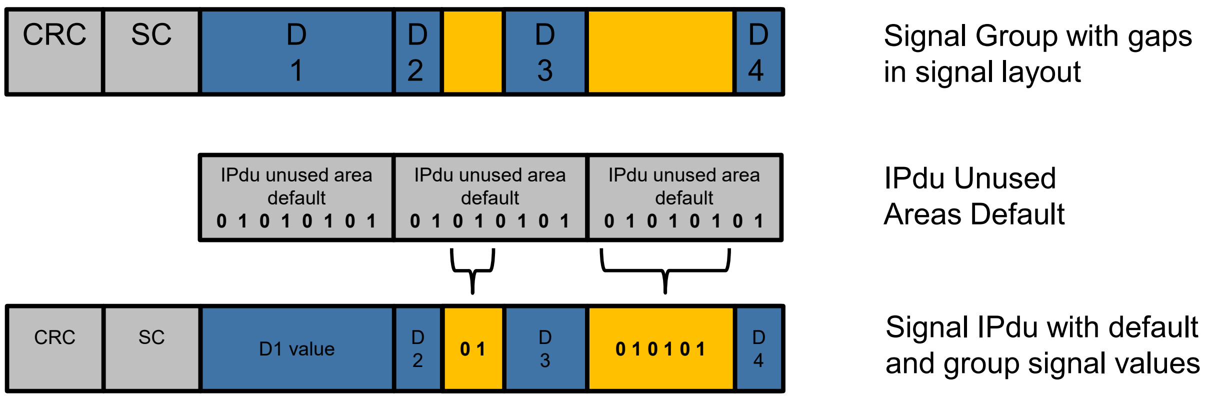
CRC
SC
D 1
D 2
D 3
D 4
Signal Group with gaps in signal layout
IPdu unused area default 0 1 0 1 0 1 0 1
IPdu unused area default 0 1 0 1 0 1 0 1
IPdu unused area default 0 1 0 1 0 1 0 1
IPdu Unused Areas Default
CRC
SC
D1 value
D 2
0 1
D 3
0 1 0 1 0 1
D 4
Signal IPdu with default and group signal values

Figure 7.4: Example of an array representation with gaps

[SWS_ComXf_00037] Buffer reservation for further transformers

Upstream requirements: SRS_Xfrm_00201, SRS_Xfrm_00202

⌈The COM Based Transformer shall consider the header and/or trailer `ISignal`s defined in the `ISignalGroup` which are processed by further transformers.⌋

[SWS_ComXf_00020]

Upstream requirements: SRS_Xfrm_00201, SRS_Xfrm_00202

⌈The COM Based Transformer shall place the serialized data element into the data buffer at the bit position according to the configuration of the `ISignalGroup` in Com.⌋

[SWS_ComXf_00013]

Upstream requirements: SRS_Xfrm_00201

⌈To allow migration, the deserialization shall be able to accept larger array representations and ignore dataElements appended at the end of a previously known parameter list.⌋

This means: data elements that were not defined in the interface specification used to generate or parameterize the deserialization code at the end of the serialized data will be ignored by the deserialization.

Class	**TransformationTechnology**			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	A TransformationTechnology is a transformer inside a transformer chain. **Tags:** xml.namePlural=TRANSFORMATION-TECHNOLOGIES			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DataTransformationSet.transformationTechnology			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
bufferProperties	BufferProperties	0..1	aggr	Aggregation of the mandatory BufferProperties.

▽

△

Class	**TransformationTechnology**			
hasInternal State	Boolean	0..1	attr	This attribute defines whether the Transformer has an internal state or not.
needsOriginal Data	Boolean	0..1	attr	Specifies whether this transformer gets access to the SWC's original data.
protocol	String	0..1	attr	Specifies the protocol that is implemented by this transformer.
transformation Description	Transformation Description	0..1	aggr	A transformer can be configured with transformer specific parameters which are represented by the Transformer Description. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=transformationDescription, transformation Description.variationPoint.shortLabel vh.latestBindingTime=postBuild
transformer Class	TransformerClassEnum	0..1	attr	Specifies to which transformer class this transformer belongs.
version	String	0..1	attr	Version of the implemented protocol.

Table 7.1: TransformationTechnology

Enumeration	**TransformerClassEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer
Note	Specifies the transformer class of a transformer.
Aggregated by	TransformationTechnology.transformerClass
Literal	***Description***
custom	The transformer is a custom transformer. **Tags:** atp.EnumerationLiteralIndex=0
safety	The transformer is a safety transformer. **Tags:** atp.EnumerationLiteralIndex=1
security	The transformer is a security transformer. **Tags:** atp.EnumerationLiteralIndex=2
serializer	The transformer is a serializing transformer. **Tags:** atp.EnumerationLiteralIndex=3

Table 7.2: TransformerClassEnum

Class	**BufferProperties**			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	Configuration of the buffer properties the transformer needs to work.			
Base	*ARObject*			
Aggregated by	TransformationTechnology.bufferProperties			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
headerLength	Integer	0..1	attr	Defines the length of the header (in bits) this transformer will add in front of the data.
inPlace	Boolean	0..1	attr	If set, the transformer uses the input buffer as output buffer.

Table 7.3: BufferProperties

Class	***TransformationDescription*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	The TransformationDescription is the abstract class that can be used by specific transformers to add transformer specific properties.			
Base	*ARObject*, *Describable*			
Subclasses	EndToEndTransformationDescription, SOMEIPTransformationDescription, UserDefinedTransformation Description			
Aggregated by	TransformationTechnology.transformationDescription			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 7.4: TransformationDescription

7.2 Error classification

Section 7.2 "Error Handling" of the document "General Specification of Basic Software Modules" [8] describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

7.2.1 Development Errors

[SWS_ComXf_00028] Definiton of development errors in module ComXf

Upstream requirements: SRS_BSW_00337

⌈

Type of error	*Related error code*	*Error value*
Error code if any other API service, except Get VersionInfo, is called before the transformer module was initialized with Init or after a call to De Init.	<MIP>_E_UNINIT	0x01
Error code if an invalid configuration set was selected	<MIP>_E_INIT_FAILED	0x02
API service called with wrong parameter	<MIP>_E_PARAM	0x03
API service called with invalid pointer	<MIP>_E_PARAM_POINTER	0x04

⌋

7.2.2 Runtime Errors

There are no runtime errors.

7.2.3 Production Errors

There are no production errors.

7.2.4 Extended Production Errors

There are no extended production errors.

8 API specification

8.1 Imported types

In the Module Interlink Headers file which is imported by the COM Based Transformer, all `ImplementationDataType`s known to the RTE are included. Using this mechanism, the COM Based Transformer knows all data types of data which shall be transformed.

[SWS_ComXf_91000] Definition of imported datatypes of module ComXf

Upstream requirements: SRS_BSW_00402

⌈

Module	***Header File***	***Imported Type***
Std	Std_Types.h	Std_VersionInfoType

⌋

8.2 Type definitions

[SWS_ComXf_00030] Definition of datatype ComXf_ConfigType

Upstream requirements: SRS_BSW_00404, SRS_BSW_00441

⌈

Name	ComXf_ConfigType	
Kind	Structure	
Elements	implementation specific	
	Type	–
	Comment	–
Description	This is the type of the data structure containing the initialization data for the transformer.	
Available via	ComXf.h	

⌋

8.3 Function definitions

The COM Based Transformer provides the specific interfaces generally required by [2].

[SWS_ComXf_00006]

Upstream requirements: SRS_Xfrm_00011

⌈The COM Based Transformer shall only provide functions for transformers where the `TransformationTechnology` is referenced as the first reference in the list of ordered references `transformerChain` from a `DataTransformation` to a `TransformationTechnology`.⌋

That means, only the first transformer in a transformer chain can be a COM Based Transformer because serializer transformer are in general only allowed to be the first transformer in a chain.

8.3.1 ComXf_<transformerId>

[SWS_ComXf_00007] Definition of API function ComXf_<transformerId>

Upstream requirements: SRS_Xfrm_00201

⌈

Service Name	ComXf_<transformerId>	

Syntax	`uint8 ComXf_<transformerId> (`
	`  uint8* buffer,`
	`  uint32* bufferLength,`
	`  <paramtype> dataElement`
	`)`	

Service ID [hex]	0x03	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	dataElement	Data element which shall be transformed
Parameters (inout)	None	
Parameters (out)	buffer	Buffer allocated by the RTE, where the transformed data has to be stored by the transformer
	bufferLength	Used length of the buffer
Return value	uint8	0x00 (`E_OK`): Serialization successful 0x81 (`E_SER_GENERIC_ERROR`): A generic error occurred
Description	This function transforms a Sender/Receiver communication using the serialization of COM Based Transformer. It takes the data element as input and outputs a uint8 array containing the serialized data.	
Available via	ComXf.h	

⌋

where

- `type` is data type of the data element after all data conversion activities of the RTE
- `paramtype` is derived from `type` according to the parameter passing rules rules defined by the *SRS BSW General* [5] (see [SRS_BSW_00484], [SRS_BSW_-

00485], and [SRS_BSW_00486]) and *SWS BSW General* [8] (see [SWS_BSW_-00186])

- transformerId is the name pattern for the transformer specified be the *General Specification on Transformers* [2] [SWS_Xfrm_00062].

The function specified in [SWS_ComXf_00007] exists for each transformed Sender/Receiver communication which uses the COM Based Transformer.

[SWS_ComXf_00008]

Upstream requirements: SRS_Xfrm_00201

⌈The function specified in [SWS_ComXf_00007] shall exist for the first reference in the list of ordered references transformerChain from a DataTransformation to a TransformationTechnology if the DataTransformation is referenced by an ISignalGroup in the role comBasedSignalGroupTransformation where the ISignalGroup references a SystemSignalGroup which is referenced by SenderReceiverToSignalGroupMapping.⌋

[SWS_ComXf_00009]

Upstream requirements: SRS_Xfrm_00201

⌈The function specified in [SWS_ComXf_00007] shall serialize complex data elements of Sender/Receiver communication into a linear byte array representation using the COM Based Transformation.⌋

8.3.2 ComXf_Inv_<transformerId>

[SWS_ComXf_00010] Definition of API function ComXf_Inv_<transformerId>

Upstream requirements: SRS_Xfrm_00201

⌈

Service Name	ComXf_Inv_<transformerId>	

Syntax	`uint8 ComXf_Inv_<transformerId> (`
	`  const uint8* buffer,`
	`  uint32 bufferLength,`
	`  <type>* dataElement`
	`)`	

Service ID [hex]	0x04	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	buffer	Buffer allocated by the RTE, where the still serialized data are stored by the Rte
	bufferLength	Used length of the buffer

▽

△

Parameters (inout)	None	
Parameters (out)	dataElement	Data element which is the result of the transformation and contains the deserialized data element
Return value	uint8	0x00 (E_OK): Serialization successful 0x01 (E_NO_DATA): No data available which can be deserialized 0x81 (E_SER_GENERIC_ERROR): A generic error occurred
Description	This function deserializes a Sender/Receiver communication using the deserialization of COM Based Transformer. It takes the uint8 array containing the serialized data as input and outputs the original data element which will be passed to the Rte.	
Available via	ComXf.h	

⌋

where

- type is data type of the data element before all data conversion activities of the RTE
- transformerId is the name pattern for the transformer specified in [SWS_-Xfrm_00062] ([2]).

The function specified in [SWS_ComXf_00010] exists for each transformed Sender/Receiver communication which uses the COM Based Transformation.

[SWS_ComXf_00011]

Upstream requirements: SRS_Xfrm_00201

⌈The function specified in [SWS_ComXf_00010] shall exist for the first reference in the list of ordered references transformerChain from a DataTransformation to a TransformationTechnology if the DataTransformation is referenced by an ISignalGroup in the role comBasedSignalGroupTransformation where the ISignalGroup references a SystemSignalGroup which is referenced by Sender-ReceiverToSignalGroupMapping.⌋

[SWS_ComXf_00035]

Upstream requirements: SRS_Xfrm_00201

⌈If ComXf_Inv_<transformerId> specified in [SWS_ComXf_00010] is called with buffer equal to NULL_PTR and bufferLength equal to 0, then the output buffer buffer shall not be changed and ComXf_Inv_<transformerId> shall return with E_NO_DATA.⌋

[SWS_ComXf_00012]

Upstream requirements: SRS_Xfrm_00201

⌈The function specified in [SWS_ComXf_00010] shall deserialize a linear byte array to primitive or complex data elements of Sender/Receiver communication using the COM Based Transformation.⌋

8.3.3 ComXf_Init

[SWS_ComXf_00026] Definition of API function ComXf_Init

Upstream requirements: SRS_BSW_00407, SRS_BSW_00411

⌈

Service Name	ComXf_Init	
Syntax	`void ComXf_Init ( const ComXf_ConfigType* config )`	
Service ID [hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	config	Pointer to the transformer's configuration data.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This service initializes the transformer for the further processing.	
Available via	ComXf.h	

⌋

8.3.4 ComXf_DeInit

[SWS_ComXf_00027] Definition of API function ComXf_DeInit

Upstream requirements: SRS_BSW_00407, SRS_BSW_00411

⌈

Service Name	ComXf_DeInit
Syntax	`void ComXf_DeInit ( void )`
Service ID [hex]	0x02
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	This service deinitializes the transformer.
Available via	ComXf.h

⌋

8.3.5 ComXf_GetVersionInfo

[SWS_ComXf_00024] Definition of API function ComXf_GetVersionInfo
Upstream requirements: SRS_BSW_00407, SRS_BSW_00411

⌈

Service Name	ComXf_GetVersionInfo	
Syntax	`void ComXf_GetVersionInfo ( Std_VersionInfoType* VersionInfo )`	
Service ID [hex]	0x00	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	VersionInfo	Pointer to where to store the version information of this module.
Return value	None	
Description	This service returns the version information of the called transformer module.	
Available via	ComXf.h	

⌋

8.4 Callback notifications

COM Based Transformer has no callback notifications.

8.5 Scheduled functions

COM Based Transformer has no scheduled functions.

8.6 Expected interfaces

COM Based Transformer has no expected interfaces.

9 Sequence diagrams

There are no sequence diagrams applicable to COM Based Transformer.

10 Configuration specification

[SWS_ComXf_00031]

Upstream requirements: SRS_Xfrm_00202

⌈The COM Based Transformer is configured based on the COM module configuration [4].⌋

Still, there is an EcuC necessary to map the implementation of the transformer. The EcuC defined in [2] shall be used.

[SWS_ComXf_00033]

Upstream requirements: SRS_Xfrm_00202

⌈The vendor specific module definition of the COM Based Transformer - based on the Xfrm configuration [2] - may be extended by the vendor to support the close interaction with the Com module [4].⌋

[SWS_ComXf_00034]

Upstream requirements: SRS_Xfrm_00202

⌈The COM Based Transformer shall be configured to be `postBuild` when the configuration of the Com module [4] is `postBuild` for the respective `ComSignalGroup`s.⌋

[SWS_ComXf_00025]

Upstream requirements: SRS_BSW_00159, SRS_Xfrm_00202

⌈The `apiServicePrefix` of the COM Based Transformer's EcuC shall be set to `ComXf`.⌋

A Referenced Meta Classes

For the sake of completeness, this chapter contains a set of class tables representing meta-classes mentioned in the context of this document but which are not contained directly in the scope of describing specific meta-model semantics.

Class	**DataTransformation**			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	A DataTransformation represents a transformer chain. It is an ordered list of transformers.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DataTransformationSet.dataTransformation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
data Transformation Kind	DataTransformationKind Enum	0..1	attr	This attribute controls the kind of DataTransformation to be applied.
executeDespite Data Unavailability	Boolean	0..1	attr	Specifies whether the transformer chain is executed even if no input data are available.
transformer Chain (ordered)	Transformation Technology	*	ref	This attribute represents the definition of a chain of transformers that are supposed to be executed according to the order of being referenced from DataTransformation.

Table A.1: DataTransformation

Class	**EcucModuleDef**			
Package	M2::AUTOSARTemplates::ECUCParameterDefTemplate			
Note	Used as the top-level element for configuration definition for Software Modules, including BSW and RTE as well as ECU Infrastructure. **Tags:** atp.recommendedPackage=EcucModuleDefs			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpDefinition*, *CollectableElement*, *Ecuc DefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
apiServicePrefix	CIdentifier	0..1	attr	For modules where several instances of the VSMD can be defined the apiServicePrefix defines the API namespace of the derived instances, e.g. Cdd, Xfrm (ComXf, SomeIpXf, E2EXf).
container	EcucContainerDef	*	aggr	Aggregates the top-level container definitions of this specific module definition. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=container.shortName xml.sequenceOffset=11
postBuildVariant Support	Boolean	0..1	attr	Indicates if a module supports different post-build variants (previously known as post-build selectable configuration sets). TRUE means yes, FALSE means no.

▽

△

Class	EcucModuleDef			
refinedModule Def	EcucModuleDef	0..1	ref	Optional reference from the Vendor Specific Module Definition to the Standardized Module Definition it refines. In case this EcucModuleDef has the category STANDARDIZED_MODULE_DEFINITION this reference shall not be provided. In case this EcucModuleDef has the category VENDOR_SPECIFIC_MODULE_ DEFINITION this reference is mandatory. **Stereotypes:** atpUriDef
supported ConfigVariant	EcucConfiguration VariantEnum	*	attr	Specifies which ConfigurationVariants are supported by this software module. This attribute is optional if the Ecuc ModuleDef has the category STANDARDIZED_ MODULE_DEFINITION. If the category attribute of the EcucModuleDef is set to VENDOR_SPECIFIC_ MODULE_DEFINITION then this attribute is mandatory.

Table A.2: EcucModuleDef

Class	**ISignal**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	Signal of the Interaction Layer. The RTE supports a "signal fan-out" where the same System Signal is sent in different SignalIPdus to multiple receivers. To support the RTE "signal fan-out" each SignalIPdu contains ISignals. If the same System Signal is to be mapped into several SignalIPdus there is one ISignal needed for each ISignalToIPduMapping. ISignals describe the Interface between the Precompile configured RTE and the potentially Postbuild configured Com Stack (see ECUC Parameter Mapping). In case of the SystemSignalGroup an ISignal shall be created for each SystemSignal contained in the SystemSignalGroup. **Tags:** atp.recommendedPackage=ISignals			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
data Transformation	DataTransformation	0..1	ref	Optional reference to a DataTransformation which represents the transformer chain that is used to transform the data that shall be placed inside this ISignal. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataTransformation.dataTransformation, dataTransformation.variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime
dataTypePolicy	DataTypePolicyEnum	0..1	attr	With the aggregation of SwDataDefProps an ISignal specifies how it is represented on the network. This representation follows a particular policy. Note that this causes some redundancy which is intended and can be used to support flexible development methodology as well as subsequent integrity checks. If the policy "networkRepresentationFromComSpec" is chosen the network representation from the ComSpec that is aggregated by the PortPrototype shall be used. If the "override" policy is chosen the requirements specified in the PortInterface and in the ComSpec are not fulfilled by the networkRepresentationProps. In case the System Description doesn't use a complete Software Component Description (VFB View) the "legacy" policy can be chosen.

▽

△

Class	**ISignal**			
initValue	ValueSpecification	0..1	aggr	Optional definition of a ISignal's initValue in case the System Description doesn't use a complete Software Component Description (VFB View). This supports the inclusion of legacy system signals. This value can be used to configure the Signal's "Init Value". If a full DataMapping exist for the SystemSignal this information may be available from a configured Sender ComSpec and ReceiverComSpec. In this case the initvalues in SenderComSpec and/or ReceiverComSpec override this optional value specification. Further restrictions apply from the RTE specification.
iSignalProps	ISignalProps	0..1	aggr	Additional optional ISignal properties that may be stored in different files. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=iSignalProps
iSignalType	ISignalTypeEnum	0..1	attr	This attribute defines whether this iSignal is an array that results in a UINT8_N / UINT8_DYN ComSignalType in the COM configuration or a primitive type.
length	UnlimitedInteger	0..1	attr	Size of the signal in bits. The size needs to be derived from the mapped VariableDataPrototype according to the mapping of primitive DataTypes to BaseTypes as used in the RTE. Indicates maximum size for dynamic length signals. The ISignal length of zero bits is allowed.
network Representation Props	SwDataDefProps	0..1	aggr	Specification of the actual network representation. The usage of SwDataDefProps for this purpose is restricted to the attributes compuMethod and baseType. The optional baseType attributes "memAllignment" and "byteOrder" shall not be used. The attribute "dataTypePolicy" in the SystemTemplate element defines whether this network representation shall be ignored and the information shall be taken over from the network representation of the ComSpec. If "override" is chosen by the system integrator the network representation can violate against the requirements defined in the PortInterface and in the network representation of the ComSpec. In case that the System Description doesn't use a complete Software Component Description (VFB View) this element is used to configure "ComSignalDataInvalid Value" and the Data Semantics. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=networkRepresentationProps
reception DefaultValue (ordered)	ValueSpecification	*	aggr	Value used to fill data on the receiver side, if less then expected data is received. The value is expected to cover the entire expected ISignal network payload.
systemSignal	SystemSignal	0..1	ref	Reference to the System Signal that is supposed to be transmitted in the ISignal.
timeout Substitution Value	ValueSpecification	0..1	aggr	Defines and enables the ComTimeoutSubstituition for this ISignal.

▽

△

Class	ISignal			
transformation ISignalProps	TransformationISignal Props	*	aggr	A transformer chain consists of an ordered list of transformers. The ISignal specific configuration properties for each transformer are defined in the TransformationISignalProps class. The transformer configuration properties that are common for all ISignals are described in the TransformationTechnology class. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=transformationISignalProps

Table A.3: ISignal

Class	ISignalGroup			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	SignalGroup of the Interaction Layer. The RTE supports a "signal fan-out" where the same System Signal Group is sent in different SignalIPdus to multiple receivers. An ISignalGroup refers to a set of ISignals that shall always be kept together. A ISignalGroup represents a COM Signal Group. Therefore it is recommended to put the ISignalGroup in the same Package as ISignals (see atp.recommendedPackage) **Tags:** atp.recommendedPackage=ISignalGroup			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
comBased SignalGroup Transformation	DataTransformation	0..1	ref	Optional reference to a DataTransformation which represents the transformer chain that is used to transform the data that shall be placed inside this ISignalGroup based on the COMBasedTransformer approach. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=comBasedSignalGroupTransformation.data Transformation, comBasedSignalGroup Transformation.variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime
iSignal	ISignal	*	ref	Reference to a set of ISignals that shall always be kept together.
systemSignal Group	SystemSignalGroup	0..1	ref	Reference to the SystemSignalGroup that is defined on VFB level and that is supposed to be transmitted in the ISignalGroup.
transformation ISignalProps	TransformationISignal Props	*	aggr	A transformer chain consists of an ordered list of transformers. The ISignalGroup specific configuration properties for each transformer are defined in the TransformationISignalProps class. The transformer configuration properties that are common for all ISignal Groups are described in the TransformationTechnology class. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=transformationISignalProps

Table A.4: ISignalGroup

Class	ISignalIPdu			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	Represents the IPdus handled by Com. The ISignalIPdu assembled and disassembled in AUTOSAR COM consists of one or more signals. In case no multiplexing is performed this IPdu is routed to/from the Interface Layer. A maximum of one dynamic length signal per IPdu is allowed. **Tags:** atp.recommendedPackage=Pdus			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *IPdu*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Pdu*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
iPduTiming Specification	IPduTiming	0..1	aggr	Timing specification for Com IPdus (Transmission Modes). This information is mandatory for the sender in a System Extract. This information may be omitted on receivers in a System Extract. atpVariation: The timing of a Pdu can vary. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=iPduTimingSpecification, iPduTiming Specification.variationPoint.shortLabel vh.latestBindingTime=postBuild
iSignalToPdu Mapping	ISignalToIPduMapping	*	aggr	Definition of SignalToIPduMappings included in the Signal IPdu. atpVariation: The content of a PDU can be variable. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=iSignalToPduMapping.shortName, iSignalTo PduMapping.variationPoint.shortLabel vh.latestBindingTime=postBuild
unusedBit Pattern	Integer	0..1	attr	AUTOSAR COM and AUTOSAR IPDUM are filling not used areas of an IPDU with this bit-pattern. This attribute is mandatory to avoid undefined behavior. This byte-pattern will be repeated throughout the IPdu.

Table A.5: ISignalIPdu

Class	ISignalToIPduMapping			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	An ISignalToIPduMapping describes the mapping of ISignals to ISignalIPdus and defines the position of the ISignal within an ISignalIPdu.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ISignalIPdu.iSignalToPduMapping, NmPdu.iSignalToIPduMapping			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
iSignal	ISignal	0..1	ref	Reference to a ISignal that is mapped into the ISignal IPdu. Each ISignal contained in the ISignalGroup shall be mapped into an IPdu by an own ISignalToIPduMapping. The references to the ISignal and to the ISignalGroup in an ISignalToIPduMapping are mutually exclusive.

▽

△

Class	**ISignalToIPduMapping**			
iSignalGroup	ISignalGroup	0..1	ref	Reference to an ISignalGroup that is mapped into the SignalIPdu. If an ISignalToIPduMapping for an ISignal Group is defined, only the UpdateIndicationBitPosition and the transferProperty is relevant. The startPosition and the packingByteOrder shall be ignored. Each ISignal contained in the ISignalGroup shall be mapped into an IPdu by an own ISignalToIPduMapping. The references to the ISignal and to the ISignalGroup in an ISignalToIPduMapping are mutually exclusive.
packingByte Order	ByteOrderEnum	0..1	attr	This parameter defines the order of the bytes of the signal and the packing into the SignalIPdu. The byte ordering "Little Endian" (MostSignificantByteLast), "Big Endian" (MostSignificantByteFirst) and "Opaque" can be selected. For opaque data endianness conversion shall be configured to Opaque. The value of this attribute impacts the absolute position of the signal into the SignalIPdu (see the startPosition attribute description). For an ISignalGroup the packingByteOrder is irrelevant and shall be ignored.
startPosition	UnlimitedInteger	0..1	attr	This parameter is necessary to describe the bitposition of a signal within an SignalIPdu. It denotes the least significant bit for "Little Endian" and the most significant bit for "Big Endian" packed signals within the IPdu (see the description of the packingByteOrder attribute). In AUTOSAR the bit counting is always set to "sawtooth" and the bit order is set to "Decreasing". The bit counting in byte 0 starts with bit 0 (least significant bit). The most significant bit in byte 0 is bit 7. Please note that the way the bytes will be actually sent on the bus does not impact this representation: they will always be seen by the software as a byte array. If a mapping for the ISignalGroup is defined, this attribute is irrelevant and shall be ignored.
transferProperty	TransferPropertyEnum	0..1	attr	Defines how the referenced ISignal contributes to the send triggering of the ISignalIPdu.
update IndicationBit Position	UnlimitedInteger	0..1	attr	The UpdateIndicationBit indicates to the receivers that the signal (or the signal group) was updated by the sender. Length is always one bit. The UpdateIndicationBitPosition attribute describes the position of the update bit within the SignalIPdu. For Signals of a ISignalGroup this attribute is irrelevant and shall be ignored. Note that the exact bit position of the updateIndicationBit Position is linked to the value of the attribute packingByte Order because the method of finding the bit position is different for the values mostSignificantByteFirst and most SignificantByteLast. This means that if the value of packingByteOrder is changed while the value of update IndicationBitPosition remains unchanged the exact bit position of updateIndicationBitPosition within the enclosing ISignalIPdu still undergoes a change. This attribute denotes the least significant bit for "Little Endian" and the most significant bit for "Big Endian" packed signals within the IPdu (see the description of the packingByteOrder attribute). In AUTOSAR the bit counting is always set to "sawtooth" and the bit order is set to "Decreasing". The bit counting in byte 0 starts with bit 0 (least significant bit). The most significant bit in byte 0 is bit 7.

Table A.6: ISignalToIPduMapping

Class	**ImplementationDataType**			
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
Note	Describes a reusable data type on the implementation level. This will typically correspond to a typedef in C-code. **Tags:** atp.recommendedPackage=ImplementationDataTypes			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dynamicArray SizeProfile	String	0..1	attr	Specifies the profile which the array will follow in case this data type is a variable size array.
isStructWith Optional Element	Boolean	0..1	attr	This attribute is only valid if the attribute category is set to STRUCTURE. If set to true, this attribute indicates that the ImplementationDataType has been created with the intention to define at least one element of the structure as optional.
subElement (ordered)	ImplementationData TypeElement	*	aggr	Specifies an element of an array, struct, or union data type. The aggregation of ImplementionDataTypeElement is subject to variability with the purpose to support the conditional existence of elements inside a Implementation DataType representing a structure. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=subElement.shortName, sub Element.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the Implementation DataType. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=symbolProps.shortName
typeEmitter	NameToken	0..1	attr	This attribute is used to control which part of the AUTOSAR toolchain is supposed to trigger data type definitions.

Table A.7: ImplementationDataType

Class	***PortPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.			
Base	*ARObject*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clientServer Annotation	ClientServerAnnotation	*	aggr	Annotation of this PortPrototype with respect to client/ server communication.
delegatedPort Annotation	DelegatedPort Annotation	0..1	aggr	Annotations on this delegated port.

▽

△

Class	*PortPrototype* (abstract)			
ioHwAbstraction Server Annotation	IoHwAbstractionServer Annotation	*	aggr	Annotations on this IO Hardware Abstraction port.
modePort Annotation	ModePortAnnotation	*	aggr	Annotations on this mode port.
nvDataPort Annotation	NvDataPortAnnotation	*	aggr	Annotations on this non voilatile data port.
parameterPort Annotation	ParameterPort Annotation	*	aggr	Annotations on this parameter port.
senderReceiver Annotation	SenderReceiver Annotation	*	aggr	Collection of annotations of this ports sender/receiver communication.
triggerPort Annotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port.

Table A.8: PortPrototype

Class	**SenderReceiverInterface**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	A sender/receiver interface declares a number of data elements to be sent and received. **Tags:** atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DataInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
dataElement	VariableDataPrototype	*	aggr	The data elements of this SenderReceiverInterface.
invalidation Policy	InvalidationPolicy	*	aggr	InvalidationPolicy for a particular dataElement
metaDataItem Set	MetaDataItemSet	*	aggr	This aggregation defines fixed sets of meta-data items associated with dataElements of the enclosing Sender ReceiverInterface

Table A.9: SenderReceiverInterface

Class	**SenderReceiverToSignalGroupMapping**			
Package	M2::AUTOSARTemplates::SystemTemplate::DataMapping			
Note	Mapping of a sender receiver communication data element with a composite datatype to a signal group.			
Base	*ARObject*, *DataMapping*			
Aggregated by	SystemMapping.dataMapping			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
dataElement	VariableDataPrototype	0..1	iref	Reference to a data element with a composite datatype which is mapped to a signal group. **InstanceRef implemented by:** VariableDataPrototypeIn SystemInstanceRef
signalGroup	SystemSignalGroup	0..1	ref	Reference to the signal group, which contain all primitive datatypes of the composite type
typeMapping	SenderRecComposite TypeMapping	0..1	aggr	The CompositeTypeMapping maps the ApplicationArray Elements and ApplicationRecordElements to Signals of the SignalGroup.

Table A.10: SenderReceiverToSignalGroupMapping

Class	**SystemSignalGroup**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	A signal group refers to a set of signals that shall always be kept together. A signal group is used to guarantee the atomic transfer of AUTOSAR composite data types. The SystemSignalGroup defines a signal grouping on VFB level. On cluster level the Signal grouping is described by the ISignalGroup element. **Tags:** atp.recommendedPackage=SystemSignalGroups			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
systemSignal	SystemSignal	*	ref	Reference to a set of SystemSignals that shall always be kept together.
transforming SystemSignal	SystemSignal	0..1	ref	Optional reference to the SystemSignal which shall contain the transformed (linear) data.

Table A.11: SystemSignalGroup

B History of Constraints and Specification Items

B.1 Constraint History of this Document according to AUTOSAR R4.2.1

Initial document release.

B.2 Constraint History of this Document according to AUTOSAR R4.2.2

B.2.1 Added Specification Items in 4.2.2

[SWS_ComXf_00032]

B.2.2 Changed Specification Items in 4.2.2

[SWS_ComXf_00028]

B.2.3 Deleted Specification Items in 4.2.2

none

B.3 Constraint History of this Document according to AUTOSAR R4.3.0

B.3.1 Added Specification Items in 4.3.0

[SWS_ComXf_00033] [SWS_ComXf_00034] [SWS_ComXf_00035]

B.3.2 Changed Specification Items in 4.3.0

[SWS_ComXf_00001] [SWS_ComXf_00004] [SWS_ComXf_00006] [SWS_ComXf_-00007] [SWS_ComXf_00008] [SWS_ComXf_00010] [SWS_ComXf_00011]

B.3.3 Deleted Specification Items in 4.3.0

none

B.4 Constraint History of this Document according to AUTOSAR R4.3.1

B.4.1 Added Specification Items in 4.3.1

[SWS_ComXf_00036]

B.4.2 Changed Specification Items in 4.3.1

[SWS_ComXf_00007] [SWS_ComXf_00023]

B.4.3 Deleted Specification Items in 4.3.1

none

B.5 Constraint History of this Document according to AUTOSAR R4.4.0

B.5.1 Added Specification Items in 4.4.0

none

B.5.2 Changed Specification Items in 4.4.0

[SWS_ComXf_00007] [SWS_ComXf_00020]

B.5.3 Deleted Specification Items in 4.4.0

[SWS_ComXf_00001] [SWS_ComXf_00014]

B.6 Constraint History of this Document according to AUTOSAR R19-11

B.6.1 Added Specification Items in 19-11

none

B.6.2 Changed Specification Items in 19-11

none

B.6.3 Deleted Specification Items in 19-11

none

B.7 Constraint History of this Document according to AUTOSAR R20-11

B.7.1 Added Specification Items in R20-11

none

B.7.2 Changed Specification Items in R20-11

none

B.7.3 Deleted Specification Items in R20-11

none

B.8 Constraint History of this Document according to AUTOSAR R21-11

B.8.1 Added Specification Items in R21-11

[SWS_ComXf_00037]

B.8.2 Changed Specification Items in R21-11

none

B.8.3 Deleted Specification Items in R21-11

none

B.9 Constraint History of this Document according to AUTOSAR R22-11

B.9.1 Added Specification Items in R22-11

none

B.9.2 Changed Specification Items in R22-11

none

B.9.3 Deleted Specification Items in R22-11

none

B.10 Constraint History of this Document according to AUTOSAR R23-11

B.10.1 Added Specification Items in R23-11

none

B.10.2 Changed Specification Items in R23-11

none

B.10.3 Deleted Specification Items in R23-11

none

B.11 Constraint History of this Document according to AUTOSAR R24-11

B.11.1 Added Specification Items in R24-11

Number	Heading
[SWS_ComXf_91000]	Definition of imported datatypes of module ComXf

Table B.1: Added Specification Items in R24-11

B.11.2 Changed Specification Items in R24-11

Number	Heading
[SWS_ComXf_00026]	Definition of API function ComXf_Init
[SWS_ComXf_00027]	Definition of API function ComXf_DeInit

Table B.2: Changed Specification Items in R24-11

B.11.3 Deleted Specification Items in R24-11

Number	Heading
[ECUC_Com_00001]	Definition of EcucForeignReferenceDef ComSystemTemplateSignalGroupRef
[ECUC_Com_00002]	Definition of EcucForeignReferenceDef ComSystemTemplateSystemSignalRef
[ECUC_Com_00017]	Definition of EcucIntegerParamDef ComTxIPduUnusedAreasDefault
[ECUC_Com_00119]	Definition of EcucEnumerationParamDef ComIPduSignalProcessing
[ECUC_Com_00127]	Definition of EcucEnumerationParamDef ComSignalType
[ECUC_Com_00157]	Definition of EcucEnumerationParamDef ComSignalEndianness
[ECUC_Com_00158]	Definition of EcucIntegerParamDef ComBitSize
[ECUC_Com_00165]	Definition of EcucIntegerParamDef ComHandleId
[ECUC_Com_00170]	Definition of EcucStringParamDef ComSignalInitValue
[ECUC_Com_00175]	Definition of EcucIntegerParamDef ComIPduHandleId
[ECUC_Com_00181]	Definition of EcucFloatParamDef ComMinimumDelayTime
[ECUC_Com_00206]	Definition of EcucReferenceDef ComIPduGroupRef
[ECUC_Com_00259]	Definition of EcucIntegerParamDef ComBitPosition
[ECUC_Com_00340]	Definition of EcucParamConfContainerDef ComIPdu
[ECUC_Com_00345]	Definition of EcucParamConfContainerDef ComSignalGroup

▽

△

Number	Heading
[ECUC_Com_00387]	Definition of EcucFunctionNameDef ComIPduCallout
[ECUC_Com_00391]	Definition of EcucStringParamDef ComSignalDataInvalidValue
[ECUC_Com_00437]	Definition of EcucIntegerParamDef ComSignalLength
[ECUC_Com_00493]	Definition of EcucEnumerationParamDef ComIPduDirection
[ECUC_Com_00496]	Definition of EcucParamConfContainerDef ComTxIPdu
[ECUC_Com_00518]	Definition of EcucReferenceDef ComIPduSignalRef
[ECUC_Com_00519]	Definition of EcucReferenceDef ComIPduSignalGroupRef
[ECUC_Com_00520]	Definition of EcucParamConfContainerDef ComGroupSignal
[ECUC_Com_00560]	Definition of EcucEnumerationParamDef ComTransferProperty
[ECUC_Com_00576]	Definition of EcucEnumerationParamDef ComTxIPduClearUpdateBit
[ECUC_Com_00709]	Definition of EcucBooleanParamDef ComIPduCancellationSupport
[ECUC_Com_00711]	Definition of EcucReferenceDef ComPduIdRef
[ECUC_Com_00761]	Definition of EcucEnumerationParamDef ComIPduType
[ECUC_Com_00765]	Definition of EcucFunctionNameDef ComIPduTriggerTransmitCallout
[ECUC_Com_10003]	Definition of EcucBooleanParamDef ComSignalGroupArrayAccess
[ECUC_Com_10006]	Definition of EcucStringParamDef ComTimeoutSubstitutionValue
[ECUC_Com_10012]	Definition of EcucChoiceReferenceDef ComIPduMainFunctionRef
[ECUC_Com_10021]	Definition of EcucReferenceDef ComMainFunctionRouteSignalsRef

Table B.3: Deleted Specification Items in R24-11